RESCUE-X

IOT-BASED SMART ACCIDENT DETECTION AND

EMERGENCY DISPATCH SYSTEM

## A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **JAGANNATH A** | **(211421106105)** |
| **KAUSHIK B** | **(211421106130)** |
| **MAHENDHAR M** | **(211421106163)** |
| **PRITHIVIRAJ E** | **(211421106333)** |
|  |  |

***in partial fulfillment for the award of the degree of***

# BACHELOR OF ENGINEERING

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

****

# PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## APRIL 2025

PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

# BONAFIDE CERTIFICATE

Certified that this project report **“RESCUE-X ( IOT-BASED SMART ACCIDENT DETECTION AND EMERGENCY DISPATCH SYSTEM )”** is the bonafide work of **“JAGANNATH A (211421106105), KAUSHIK B (211421106130), MAHENDHAR M (211421106163), PRITHIVIRAJ E (21142110631)”** who carried out the project work under my supervision.

**SIGNATURE** Dr.S.RAJAKUMAR, M.E.,

Ph.D., PROFESSOR

## HEAD OF THE DEPARTMENT

Department of Electronics and Communication Engineering Panimalar Engineering College Chennai – 600 123.

**SIGNATURE**

Ms.R.MARY VICTORIA, M.E., ASSISTANT PROFESSOR **SUPERVISOR**

Department of Electronics and Communication Engineering Panimalar Engineering College Chennai – 600 123.

Certified that the above mentioned students were examined in the university project viva-voce held on

## INTERNAL EXAMINER EXTERNAL EXAMINER

# ACKNOWLEDGEMENTS

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who helped us in the completion of this project.

We would like to express our deep gratitude to Our Beloved Secretary and Correspondent, **Dr.P.Chinnadurai M.A., Ph.D.,** for his motivation which inspired us a lot in completing this project work. We also offer our sincere thanks to our dynamic directors **Mrs.C.Vijayarajeswari, Dr.C.Sakthi Kumar M.E., Ph.D.,** and **Dr. Saranya Sree Sakthi Kumar B.E., M.B.A, Ph.D.,** for providing us with the necessary facilities and support needed to complete this project.

We also express our appreciation and gratefulness to our principal **Dr. K. Mani M.E., Ph.D.,**who never failed to motivate and encourage us in all our initiatives and also helped us in the completion of the project.

We would like to extend our gratitude and thanks to our Head of the Department **Dr.S.Rajakumar M.E., Ph.D.,** for his valuable suggestions and continuous encouragement throughout the completion of our project.

Our utmost gratitude and thanks to our project supervisor **Ms.R.Mary Victoria M.E,** Assistant Professor, Department of Electronics and Communication Engineering for her constant guidance, motivation, and valuable suggestions throughout the course of this project work and for helping us to complete this project work on time.

We are grateful to our beloved parents for providing us with all the support and the opportunities to complete this project work. We also express our sincere thanks to all our friends and family members for their endless support.

# ABSTRACT

The rising number of road accidents demands an efficient emergency response system to ensure prompt medical assistance. This project introduces a Real-Time Accident Detection and IoT-Based Ambulance Dispatch System designed to enhance emergency services and public safety. The system employs advanced sensors like accelerometers, gyroscopes, and impact sensors to detect sudden impacts or abnormal vehicle patterns indicative of an accident. Upon detection, the data is sent to a controller that verifies the incident to minimize false alarms. Once confirmed, GSM technology transmits crucial details such as location, time, and severity to a centralized web server. This server notifies the nearest available ambulance driver through a dedicated web interface, ensuring efficient resource allocation by preventing multiple ambulances from responding to the same incident. After accepting the request, the ambulance driver can select a hospital based on proximity or medical capability. An IoT-enabled GPS module in the ambulance enables real-time route monitoring, continuously updating the path to avoid traffic congestion and other obstacles. This dynamic system optimizes response time, ensuring accident victims receive timely medical attention. By combining sensor technology, GSM communication, and IoT-based navigation, the system provides a comprehensive and automated emergency response solution.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | vii |
|  | **LIST OF ABBREVIATIONS** | viii |
| **1** | **INTRODUCTION** | 1 |
|  | BACKGROUND AND NEED FOR THE PROJECT | 2 |
|  | PURPOSE OF THE PROJECT | 2 |
|  | SYSTEM OVERVIEW | 3 |
|  | 1. SCOPE OF THE PROJECT | 3 |
|  | 1. OBJECTIVE | 4 |
|  | PROBLEM STATMENT | 4 |
|  | MOTIVATION FOR THE PROJECT | 5 |
| **2** | **LITERATURE SURVEY** | 6 |
| **3** | **EXISTING WORK** | 12 |
|  | 1. OVERVIEW OF CURRENT ACCIDENT DETECTION AND EMERGENCY RESPONSE SYSTEM | 13 |
|  | 1. LIMITATION OF CURRENT SYSTEM | 13 |
|  | 1. DISADVANTAGES OF THE EXISTING SYSTEM | 14 |
| **4** | **PROPOSED SYSTEM** | 16 |
|  | 1. SYSTEM OVERVIEW | 17 |
|  | 1. ADVANTAGES | 17 |
|  | 1. HARDWARE COMPONENTS | 18 |
|  | 1. SOFTWARE COMPONENTS | 18 |
|  | SYSTEM CONFIGURATION | 18 |
|  | Hardware System Configuration |  |
|  | Software System Configuration |  |
|  | 1. LIST OF MODULES | 22 |
|  | 1. MODULE DESCRIPTION | 22 |
|  | 1. Accident Detection Module | 22 |
|  | 1. GSM Communication Module | 23 |
|  | 1. Ambulance Dispatch and Notification Module | 23 |
|  | 1. Route Optimization Module | 23 |
|  | 1. Web Interface for Monitoring and Control | 24 |
|  | 1. PROPOSED SYSTEM ALGORITHM | 29 |
|  | 1. SOFTWARE TESTING | 53 |
| **5** | **RESULTS AND DISCUSSION** | 56 |
| **6** | **CONCLUSION** | 60 |
|  | REFERENCES | 63 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE**  **NO.** | **TITLE** | **PAGE**  **NO.** |
| 4.3.1 | ESP 32 Micro-Controller | 18 |
| 4.3.2 | Power Supply | 18 |
| 4.3.3 | IoT (Internet of Things) | 18 |
| 4.3.4 | UART (Universal Asynchronous Receiver-Transmitter) | 19 |
| 4.3.5 | Micro Electro-Mechanical Sensor (MEMS) | 19 |
| 4.3.6 | GSM Module | 19 |
| 4.5.3 | Block Diagram | 20 |
| 4.5.4 | Architecture Diagram | 20 |
| 4.7.5 | UML Diagram (Use-Case Diagram) | 24 |
| 4.7.6 | Activity Diagram | 25 |
| 4.7.7 | Data Flow Diagram (Level 0 – Level 4) | 26 |
| 4.7.8 | Overall Diagram | 27 |
| 7 | Final Prototype | 60 |
| 8 | Notification Message | 61 |
| 9 | Web page Application | 61 |
| 10 | Experimental Graph | 62 |
| 11 | Experimental Result:   * Comparison of accident detected * Accuracy percentage of experiment * False reporting of experiments * Parameter based comparison | 62 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| GSM | – Global System for Mobile Communication |
| IoT | – Internet of Things |
| MEMS | – Micro-Electro Mechanical Systems |
| ESP | Espressif Systems Platform (Commonly refers to ESP32/ES8266 Microcontrollers) |
| GPS | Global Positioning System |
| UART | – Universal Asynchronous Receiver-Transmitter |
| ****HTML**** | – HyperText Markup Language |
| CSS | – Cascading Style Sheets |
| SMS | Short Message Server |
| LCD | Liquid Crystal Display |
| GPRS | General Packet Radio Service |

## Chapter 1

## INTRODUCTION

## CHAPTER 1

## INTRODUCTION

### ****1.1 Background and Need for the Project****

Road accidents have become a serious global concern in recent years, leading to countless fatalities and leaving many victims with life-altering injuries. The aftermath of an accident often depends heavily on how quickly emergency medical services can reach the scene and provide immediate assistance. Unfortunately, in many cases, delays in accident detection and ambulance dispatch significantly reduce the chances of survival and recovery, especially in remote or highly congested urban areas. One of the primary reasons for this delay is the lack of real-time accident detection mechanisms and efficient communication between accident sites and emergency services. Traditional emergency response systems rely heavily on eyewitnesses or passersby to report accidents, causing valuable time to be lost before help is even dispatched.

### ****1.2 Purpose of the Project****

This project is designed to address these critical gaps by developing a **Real-Time Accident Detection and Emergency Response System**. The system uses a combination of **advanced sensors, GSM technology, and IoT-based GPS tracking** to detect accidents instantly, notify nearby ambulances, and guide them efficiently to both the accident site and the nearest hospital. By automating the accident detection and ambulance dispatch process, the system ensures a rapid response, minimizes human error, and significantly reduces the time taken for emergency services to reach accident victims.

### ****1.3 System Overview****

The system architecture consists of the following core components:

* **Accident Detection Unit:** Using **MEMS sensors** and an **ESP32 microcontroller**, the system continuously monitors vehicle dynamics and impact forces. The sensors detect sudden changes like collisions or rollovers, triggering the system to initiate emergency protocols.
* **GSM Communication Module:** Once an accident is confirmed, the system utilizes GSM to send real-time alerts to a centralized web server connected to nearby ambulance drivers.
* **Ambulance Dispatch and Locking Mechanism:** To avoid confusion and multiple responses, the system allows **only the nearest available ambulance** to accept the emergency request. Once accepted, the job is locked for that ambulance, preventing duplication.
* **IoT-Enabled GPS Route Management:** The assigned ambulance driver inputs the name of the hospital, and the system calculates the fastest possible route. Live traffic data and road conditions are factored in for dynamic rerouting, ensuring optimal travel time.

### ****1.4 Scope of the Project****

The primary focus of this project is to:

* **Reduce emergency response times** by detecting accidents automatically and notifying the closest available ambulance driver.
* Provide **dynamic route updates** that factor in real-time traffic conditions and road situations, ensuring faster arrival at both the accident site and hospital.
* **Prevent resource duplication** by ensuring that only one ambulance is assigned per accident, improving operational efficiency.
* Establish a scalable and reliable system that can be integrated into **smart city infrastructures** for improved public safety.

Ultimately, the system aims to **enhance emergency medical services** by providing a well-coordinated, technologically advanced response mechanism that minimizes fatalities and injury severity resulting from road accidents.

### ****1.5 Objectives****

The key objectives of this project are:

* To design a system that **detects accidents in real-time** using MEMS sensors and an ESP32 controller.
* To ensure **instant communication** with ambulance drivers using GSM technology.
* To enable **real-time ambulance dispatch**, ensuring that only one ambulance is assigned per incident.
* To integrate **IoT-based GPS tracking** for providing ambulances with the shortest and safest route to the hospital.
* To reduce emergency response time, optimize resource allocation, and improve the overall **efficiency of accident response systems**.

### ****1.6 Problem Statement****

Timely medical intervention is crucial in accident scenarios; however, several issues plague existing systems:

* **Delayed accident detection** due to reliance on third-party reporting.
* **Ambulance dispatch inefficiencies** that lead to slower response times.
* **Poor coordination** between ambulance drivers and hospitals.
* **Static routing** that fails to account for live traffic conditions, causing further delays.
* **Resource duplication**, where multiple ambulances respond to the same incident due to lack of real-time system coordination.

These challenges compromise the effectiveness of emergency response services and can lead to preventable loss of life. Therefore, an integrated system combining sensors, GSM, GPS, and IoT is essential for improving emergency response efficiency.

### ****1.7 Motivation for the Project****

The increasing number of road accidents, rising fatalities, and preventable injuries serve as the primary motivation behind this project. Studies indicate that **golden-hour medical attention**—medical care provided within the first hour of an accident—is critical to survival.

However, existing systems are either outdated or incapable of providing fast, efficient, and organized responses. By implementing advanced sensor technology, GSM-based communication, and real-time GPS tracking, this project strives to:

* **Reduce the time gap between accident occurrence and emergency response**.
* **Save lives by ensuring faster ambulance arrival** at accident sites.
* **Improve coordination and resource management** during emergencies.
* Contribute to **public safety initiatives** and lay the groundwork for future integration into smart city infrastructures.

## Chapter 2

### LITERATURE SURVEY

## CHAPTER 2

### LITERATURE SURVEY

**PAPER 1**

**TITLE:** Fall Detection With Wrist-Worn Watch By Observations In Statistics Of Acceleration

**AUTHOR:** Songsheng Li

**YEAR:** 2023

**ABSTRACT:**

It is common for older people to live alone, which can have tragic consequences if they have an accident and can’t call for help in time. This is particularly acute in an aging society where falling is one of the most common accidents. According to the CDC, 1/4 of people over the age of 65 in the United States fall each year. The development of IoT and MEMS has made it possible to detect falls in time and automatically call for help. The presented fall detection system focuses on the walk-fall-still pattern, collects accelerations through the wrist-worn M5StickC-Plus watch, analyses the data locally in the watch, detects falls using an algorithm based on observations in the statistics of acceleration in one second, and then transmits the alarm signal to a remote healthcare system in real-time via WIFI. The lightweight algorithm has been proven to be 90% accurate in detecting falls, and the system can notify service staff of accidents within 1 second. The features of comfort, lightness, and timeliness make the device more practical than similar products. The low-cost, non-intrusive device can be used in care homes and is also suitable for elderly people living alone.

**PAPER 2**

**TITLE:** A Real-Time Vision Transformers-Based System For Enhanced Driver Drowsiness Detection And Vehicle Safety

**AUTHOR:** Anwar Jarndal, Hissam Tawfik, Ali I. Siam, Imad Alsyouf, Ali Cheaitou

**YEAR:** 2025

**ABSTRACT:**

Drowsy driving is a leading cause of fatal traffic accidents worldwide. Drowsy driving has emerged from modern societal trends such as long working hours, heavy reliance on vehicles, and insufficient sleep. Despite considerable efforts by researchers to develop efficient driver drowsiness detection systems, none so far has been widely adopted due to their high cost, intrusive nature, and ineffectiveness in challenging real-life situations. This paper presents a novel, real-time, non-intrusive, and cost-effective driver drowsiness detection system leveraging vision transformers (ViT). Our approach detects the driver’s face from each video frame and classifies the driver’s state as either ‘drowsy’ or ‘alert’ based on the entire facial image, as opposed to previous systems that rely on analyzing specific facial features. We demonstrate that the proposed Vision Transformers-based Driver Drowsiness Detection (ViT-DDD) system surpasses existing state-of-the-art methods, particularly in challenging scenarios such as drivers wearing glasses or sunglasses, or in different lighting conditions. The model was trained and evaluated on two widely used public drowsiness detection datasets, achieving classification accuracies of 98.89% on the NTHU-DDD dataset and 99.4% on the UTA-RLDD dataset. Furthermore, the system was successfully deployed on a Raspberry-Pi microcomputer, integrated with an infrared camera, a GSM/GPS module, and a buzzer to alert the driver and report the drowsiness condition to the vehicle owner.

**PAPER 3**

**TITLE:** A Comprehensive Study On Iot Based Accident Detection Systems For Smart Vehicles

**AUTHOR:** Unaiza Alvi, Muazzam A. Khan Khattak, Balawal Shabir, Asad Waqar Malik, Sher Ramzan Muhammad

**YEAR:** 2020

**ABSTRACT:**

With population growth, the demand for vehicles has increased tremendously, which has created an alarming situation in terms of traffic hazards and road accidents. The road accidents percentage is growing exponentially and so are the fatalities caused due to accidents. However, the primary cause of the increased rate of fatalities is due to the delay in emergency services. Many lives could be saved with efficient rescue services. The delay happens due to traffic congestion or unstable communication to the medical units. The implementation of automatic road accident detection systems to provide timely aid is crucial.

Many solutions have been proposed in the literature for automatic accident detection. The techniques include crash prediction using smartphones, vehicular ad-hoc networks, GPS/GSM based systems, and various machine learning techniques. With such high rates of deaths associated with road accidents, road safety is the most critical sector that demands significant exploration. In this paper, we present a critical analysis of various existing methodologies used for predicting and preventing road accidents, highlighting their strengths, limitations, and challenges that need to be addressed to ensure road safety and save valuable lives.

**PAPER 4**

**TITLE:** Smart System To Avoid Car Accidents

**AUTHOR:** Muhammed Saffarini, Rasha Saffarini, Isam Ishaq

**YEAR:** 2020

**ABSTRACT:**

Car accidents have different reasons, they are either caused by external (outside the car) or internal factors (inside the car). Accidents due to external reasons occur because of environmental reasons such as obstructed vision of the driver due to fog or bad road conditions. Internal factors include decreased amount of Oxygen, and in turn increase the amount of carbon dioxide, driver sleep, humidity and temperature ratio between outside and inside the car, which cause condensation on the front windshield which limits the vision of the driver. To avoid car accidents, one should minimize both external as well as internal accident reasons. In this paper, a system consisting of two parts is proposed to monitor the external and internal driving conditions. The first one focuses on external accident conditions, which monitors the road and notifies the driver about any problem in front of him and take action to avoid it, by reducing their speed, increase the lights of the car, or by using smart bumps that come out when needed (the risk of sliding for example) to enforce the driver to drive slowly and carefully. These smart bumps are needed also when the sensors detect that the driver feels sleepy. Activating them will reduce the probability of an accident. Additionally, if the humidity and temperature of the environments increased over a specified range, fans are activated to cool down the temperature inside the car. The other system works inside the car. If the air conditions pose a danger on the driver (the temperature, or CO2 increased for example), the system takes an action by starting the air condition or opening the windows to balance the temperature or CO2 ratios. The system will be applied by using sensors to measure factors inside and outside the car, which in turn tells the cars what to do by the interaction between both systems via the internet.

**PAPER 5**

**TITLE:** Vehicle Speed Control and Accident-Avoidance System Based on Arm M4 Microprocessor

**AUTHOR:** Waladur Rahman, Md. Raseduzzaman Ruman, Khan Roushan Jahan, Md.

Jamil Roni, Md. Foyjur Rahman, Md. Abu Hasnat Shahriar

**YEAR:** 2020

**ABSTRACT:**

Now a day’s road accident is one of the major concerns in our country. Reckless vehicle driving is the major reason behind those road accidents. Alarming rate of accidents and uncontrollable car in the road demand an automatic system that would guide drivers immediate in dangerous situation. When any obstacle (like human body, vehicle, and other object) comes in front of the vehicle, speed control of the vehicle is the viable solution to avoid accident. We propose a solution in our project to avoid road accidents and to control speed of vehicles. Front Ultra-sonic sensors of the vehicles detect obstacles, then Cortex ARM M4 process the information and passes signals to the wiper motor to brake the vehicle with the help of Arduino Uno. Furthermore, if any obstacle from the backside come closer to vehicles, immediately a buzzer will alert the driver.

Chapter 3

EXISTING SYSTEM

CHAPTER 3

EXISTING SYSTEM

### ****3.1 Overview of Current Accident Detection and Emergency Response Systems****

At present, most accident detection and emergency response systems depend heavily on **traditional methods** that require human intervention. When an accident occurs, the most common practice is for bystanders, victims, or other road users to manually report the incident by calling emergency services through dedicated helpline numbers like **108 or 911**. This method is highly unreliable, especially in remote areas or during odd hours when no one may witness the accident.

Some modern systems have attempted to automate accident detection using **basic sensors or crash detection modules** embedded in vehicles. However, these systems are often limited in functionality, only detecting the impact after it has occurred without offering any immediate or automated communication to emergency services.

### ****3.2 Limitations of Current Systems****

While these systems may work in urban environments with better infrastructure, several drawbacks limit their overall effectiveness, particularly in **reducing response time** and **ensuring optimal emergency service deployment**. Most existing systems face the following challenges:

* **Delayed Accident Reporting:**  
  Dependence on manual reporting leads to delays. If no one witnesses the accident or is unable to make the emergency call, the response time increases significantly, potentially costing lives.
* **Lack of Real-Time Communication:**  
  Existing systems do not support continuous real-time communication between the accident detection unit and emergency service providers, causing delays in dispatch and action.
* **Inefficient Dispatch System:**  
  Many emergency systems lack the capability to assign the **nearest available ambulance**. Instead, multiple ambulances may be sent, or resources may be wasted on distant responders while nearby ambulances remain idle.
* **No Route Optimization:**  
  Current systems rarely integrate **GPS or live traffic data**. As a result, ambulances are dispatched without considering traffic congestion, roadblocks, or alternative faster routes, leading to longer travel times.
* **Limited Geographic Coverage:**  
  Remote or rural areas often suffer from inadequate mobile network coverage, making it difficult for traditional systems to communicate effectively with emergency responders.

### ****3.3 Disadvantages of the Existing System****

The following points highlight the critical shortcomings of current accident detection and emergency response frameworks:

* **Communication Failures in Remote Areas:**  
  Existing systems heavily rely on human reporting or network connectivity. In rural or remote locations, the absence of reliable mobile coverage can result in complete communication failure, leaving victims without timely help.
* **Lack of Continuous Ambulance Tracking:**  
  Once dispatched, ambulances are usually not monitored in real-time. This makes it difficult for control center’s to track their progress, estimate arrival times, or reroute them in case of traffic or road closures.
* **No Real-Time Traffic Consideration:**  
  Ambulance routing is mostly static, and drivers often take the shortest physical route without considering real-time traffic conditions. In dense urban areas or during peak hours, this leads to significant delays, which could be fatal in critical.

## Chapter 4

## PROPOSED SYSTEM

CHAPTER 4

**PROPOSED SYSTEM**

**4.1 SYSTEM OVERVIEW:**

The proposed system aims to address the limitations of existing accident detection and emergency response frameworks by integrating advanced technologies such as ESP32 controllers, MEM sensors, GSM communication, and IoT for real-time accident detection, efficient ambulance dispatch, and dynamic route optimization. In this system, MEM sensors embedded in vehicles detect sudden changes in velocity or impact, instantly triggering an alert to the nearest available ambulance via GSM. This ensures that emergency responders are notified without delay, reducing response times significantly. Once the ambulance driver accepts the notification, they can input the nearest hospital name, and the system automatically updates the optimal route using IoT-enabled GPS technology. This real-time route optimization takes into account live traffic data, road conditions, and distance, ensuring the ambulance reaches the hospital in the shortest possible time. The system also includes a web-based interface where ambulance drivers can view accident notifications, monitor their progress, and interact with the system in real time.

**4.2** **ADVANTAGES:**

* By notifying the nearest ambulance instantly through GSM, the system significantly reduces response times, ensuring quicker assistance for accident victims.
* The web interface enables ambulance drivers and emergency services to monitor accident data, track ambulances, and manage responses in real time.
* The system provides real-time updates to hospitals, allowing them to prepare for incoming patients, ensuring that medical staff is ready for immediate intervention.

**4.3 HARDWARE COMPONENTS:**

* ESP32 MICRO CONTROLLER
* POWER SUPPLY
* IOT
* UART
* MEM SENSOR
* GSM

**4.4 SOFTWARE COMPONENTS:**

* EMBEDDED C
* ADRUINO IDE

**4.5** **SYSTEM CONFIGURATION:**

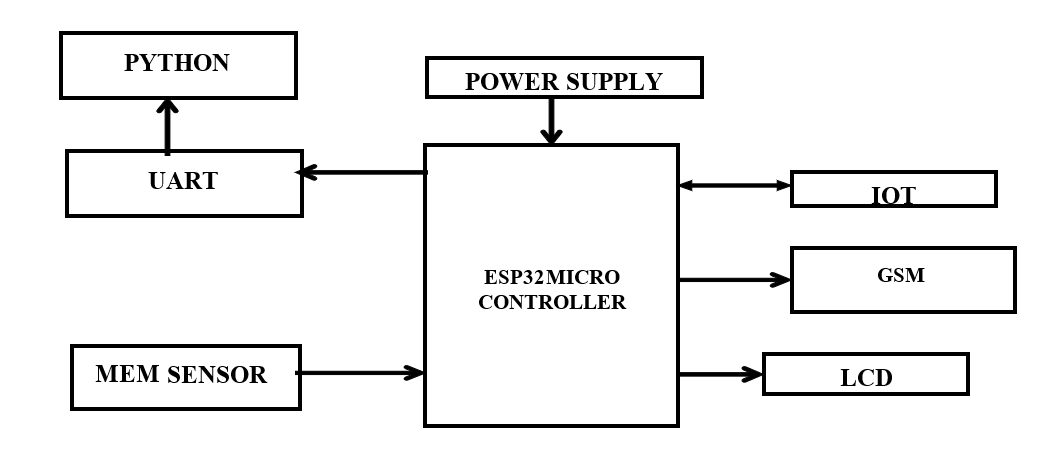
* + 1. **H/W SYSTEM CONFIGURATION**:
    - Processor – I3, i5, i7
    - RAM - 8 Gb
    - Hard Disk - 500GB
    1. **S/W SYSTEM CONFIGURATION:**
    - Operating System - Windows 8/10/11
    - Front End - Html, Css
    - Scripts – Flask (Python)
    - Tool – Python idle

**HARDWARE COMPONENTS:**

|  |  |
| --- | --- |
| **Components** | **Explanation** |
| Esp32 Arduino-esp32 Dev Module  **4.3.1 ESP32 MICRO CONTROLLER** | A powerful low-cost microcontroller with built in Wi-Fi and Bluetooth supports IoT application has multiple UART interfaces, GPIOs, ADCs and more. |
| Power supply Arduino, ESP32 and ESP8266: See your options!****4.3.2 POWER SUPPLY**** | Provides required voltage and current to all components (typically 5V or 3.3V for the ESP32) Battery or adapter-based, depending on project portability. |
| Internet of things, wireless communication network, abstract image ... **4.3.3 IOT ( INTERNET OF THINGS)** | Enables remote monitoring and control of devices over the internet. ESP32 acts as the IoT gateway, sending data to cloud platforms like Blynk , ThingSpeak , MQTT Servers and Firebase. |

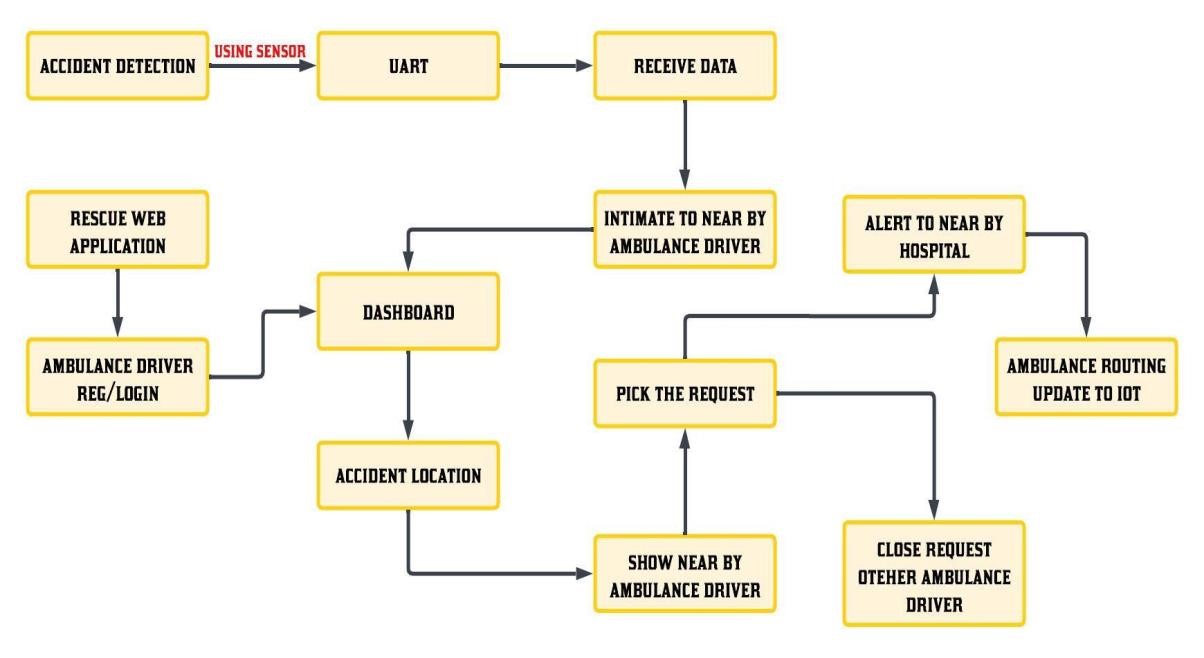
|  |  |
| --- | --- |
| **Components** | **Explanation** |
| PL2303 USB转UART 串口TTL模块V2 USB type A接口  **4.3.4 UART (Universal Asynchronous Receiver Transmitter)** | Serial communication protocol used for data exchange ESP32 uses UART to communicate with:  * + **GSM Module**   + **MEM Sensor**   + **PC (via USB)** |
| **Mems Sensor Working Principle Types Advantages Its Applications Images**  **4.3.5 Micro Electro-Mechanical Sensor (MEMS)** | A MEMS sensor is a miniature device that detects physical changes like acceleration, motion, or pressure using micro-electromechanical components. It is widely used in accident detection to sense sudden impacts or crashes. |
| Buy SIM7000A 4G GSM Module NB-IoT eMTC LTE-CAT-M1 Quad-Band FDD-LTE B2 ...   **4.3.6 IOT ( INTERNET OF THINGS)** | Enables cellular communication for SMS Alerts and Remote data access where Wi-Fi isn’t available and Connected to ESP32 via UART. |

# 4.5.3 BLOCK DIAGRAM



## The system integrates MEMS (Micro-Electro-Mechanical Systems) sensors or accelerometers to monitor sudden impacts, tilts, or vibrations, which may indicate a vehicle accident. If a crash is detected, the ESP32 microcontroller processes the sensor data and, if the impact exceeds a predefined threshold, it triggers an emergency alert. The GSM module (SIM800L or similar) sends an SMS alert containing the accident details and location to emergency contacts, such as ambulance services, police, or family members.

**4.5.4 ARCHITECTURE DIAGRAM**



**4.6 LIST OF MODULES:**

* Accident Detection Module
* GSM Communication Module
* Ambulance Dispatch and Notification Module
* Route Optimization Module
* Web Interface for Monitoring and Control

**4.7 MODULES DESCRIPTIONS:**

**4.7.1 ACCIDENT DETECTION MODULE:**

The Accident Detection Module utilizes MEM (Microelectromechanical) sensors to detect sudden impacts or abnormal changes in a vehicle’s movement, which are indicative of an accident. MEM sensors, which are highly sensitive to vibrations and accelerations, are embedded in the vehicle to continuously monitor its motion. When an accident occurs, such as a collision or sudden deceleration, the MEM sensor detects the sharp change in acceleration or velocity, triggering an immediate signal to the ESP32 controller. The controller processes this information and determines whether the detected change meets the threshold criteria for an accident.

**4.7.2** **GSM COMMUNICATION MODULE:**

Once the Accident Detection Module identifies a crash or abnormal vehicle behaviour, it triggers the GSM module to send an SMS or automated call to the nearest ambulance driver, notifying them of the accident. This communication includes vital information such as the location of the accident, the severity of the incident, and any immediate medical requirements. The GSM module ensures that even in remote areas where internet connectivity may be limited, the system can still function effectively using cellular networks.

**4.7.3 AMBULANCE DISPATCH AND NOTIFICATION MODULE:**

The system uses geographic location data (obtained through GPS) to identify the closest ambulance and sends a notification to the driver, including critical details such as the accident location, severity, and any specific instructions for the response. This notification is sent in real-time, ensuring that the ambulance driver receives the information without delay. The ambulance driver can then accept or reject the notification through a web interface, which helps prevent multiple ambulances from responding to the same accident, optimizing resource allocation.

**4.7.4 ROUTE OPTIMIZATION MODULE:**

Once the ambulance driver accepts the accident notification, they are prompted to input the nearest hospital to the accident location. This data is then used to dynamically update the ambulance's route in real-time. The system employs IoT-enabled GPS technology to calculate the most efficient path based on live traffic conditions, road closures, and other real-time data. Additionally, the system automatically updates the route to reflect any changes or detours, ensuring continuous optimization throughout the journey. Furthermore, as the ambulance progresses along the optimized route, the system notifies the local police to alert them of the ambulance's approach, allowing them to clear the path and ensure the ambulance can pass through without delays.

**4.7.5 WEB INTERFACE FOR MONITORING AND CONTROL:**

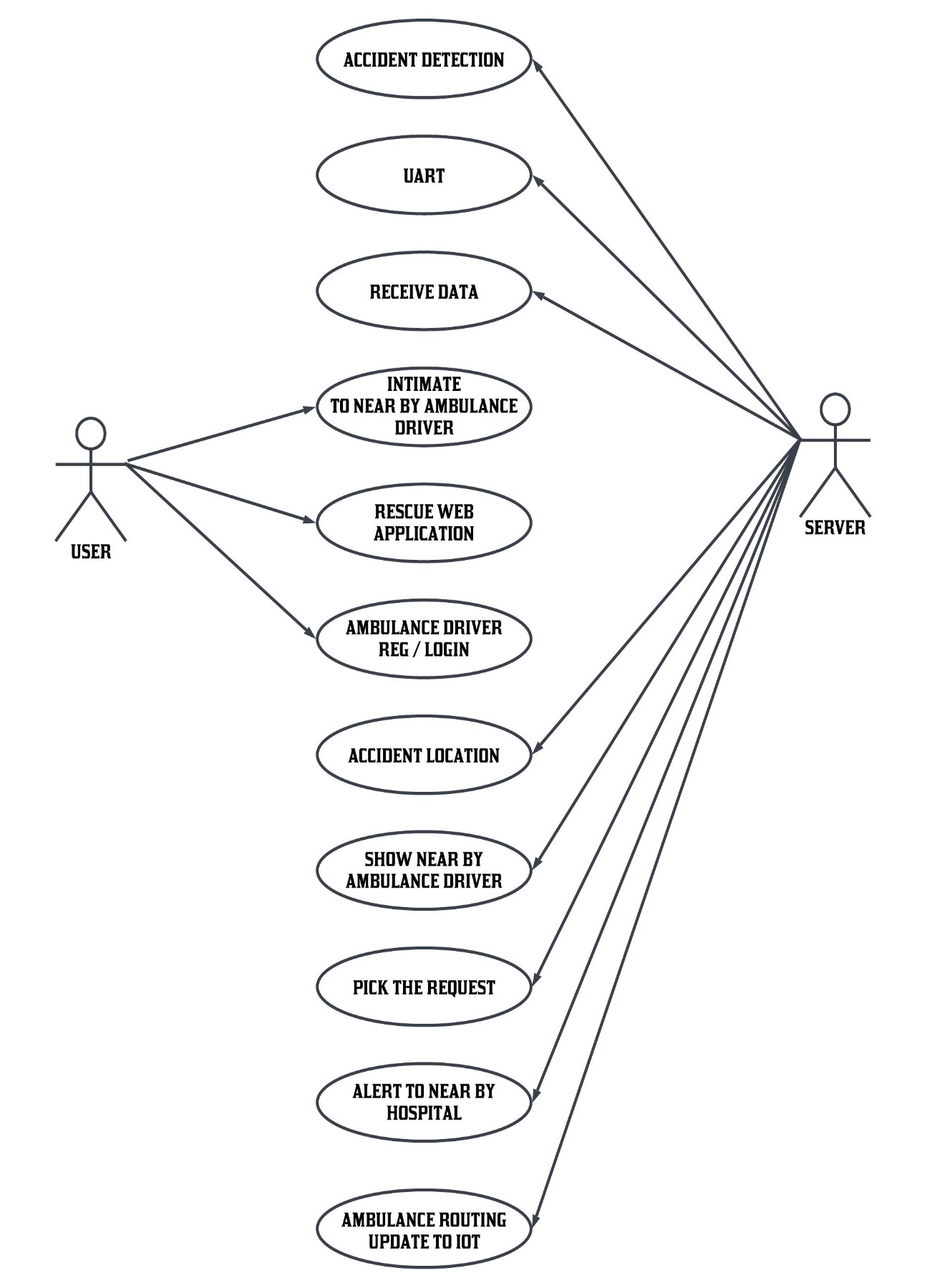
The web interface is a critical component of the proposed accident detection and ambulance rescue system, designed to provide an efficient and interactive platform for monitoring accident data and assisting emergency responders. This interface enables ambulance drivers and authorized personnel to access real-time information about accidents, ensuring faster response times and improved coordination. The web application features a secure login system, allowing ambulance drivers to log in using their credentials to access accident alerts and relevant data. Upon logging in, users are presented with a dashboard that displays key details such as accident location, GPS coordinates, and the time of occurrence.To improve navigation efficiency, the system integrates the Google Maps API, which dynamically plots accident locations on a map interface. This map view provides clear visual guidance, helping ambulance drivers identify the fastest and most efficient routes to the accident site. The map interface also includes essential features like zoom controls, route optimization, and live traffic updates, which further enhance navigation and minimize delays during emergencies.

In addition to displaying accident details, the web interface facilitates real-time data synchronization with the hardware system. Whenever an accident is detected, the ESP32 microcontroller transmits the relevant data via the GSM module to the web application, ensuring accident details are displayed promptly without delays. This seamless data integration is essential for ensuring ambulance drivers receive accurate and up-to-date information to make informed decisions during emergencies. The web interface also offers control functionalities, allowing authorized personnel to configure system settings, manage emergency contact details, and monitor system status. This feature ensures flexibility and adaptability to suit different operational requirements. Furthermore, the web interface maintains a log history of past accident alerts, providing valuable data for analysis and performance evaluation.

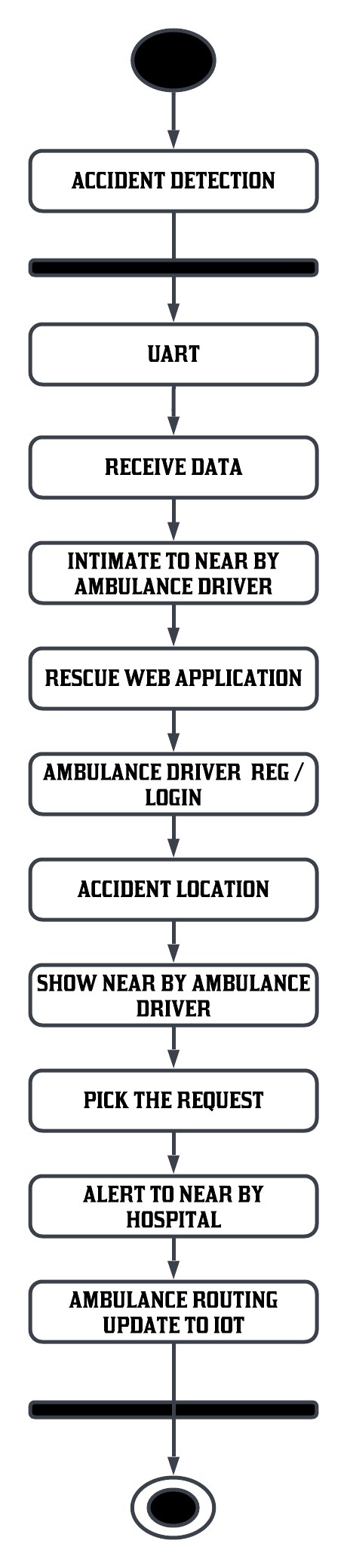
The interface is designed with an intuitive layout, ensuring ease of use even for non-technical users. Clear labelling, organized dashboards, and interactive elements allow ambulance drivers to quickly access crucial information during emergencies.

**UML DIAGRAMS:**

**USECASE DIAGRAM:**



**ACTIVITY DIAGRAM:**



**DATAFLOW DIAGRAM:**

**LEVEL: 0**

**LEVEL: 1**



**LEVEL: 2**

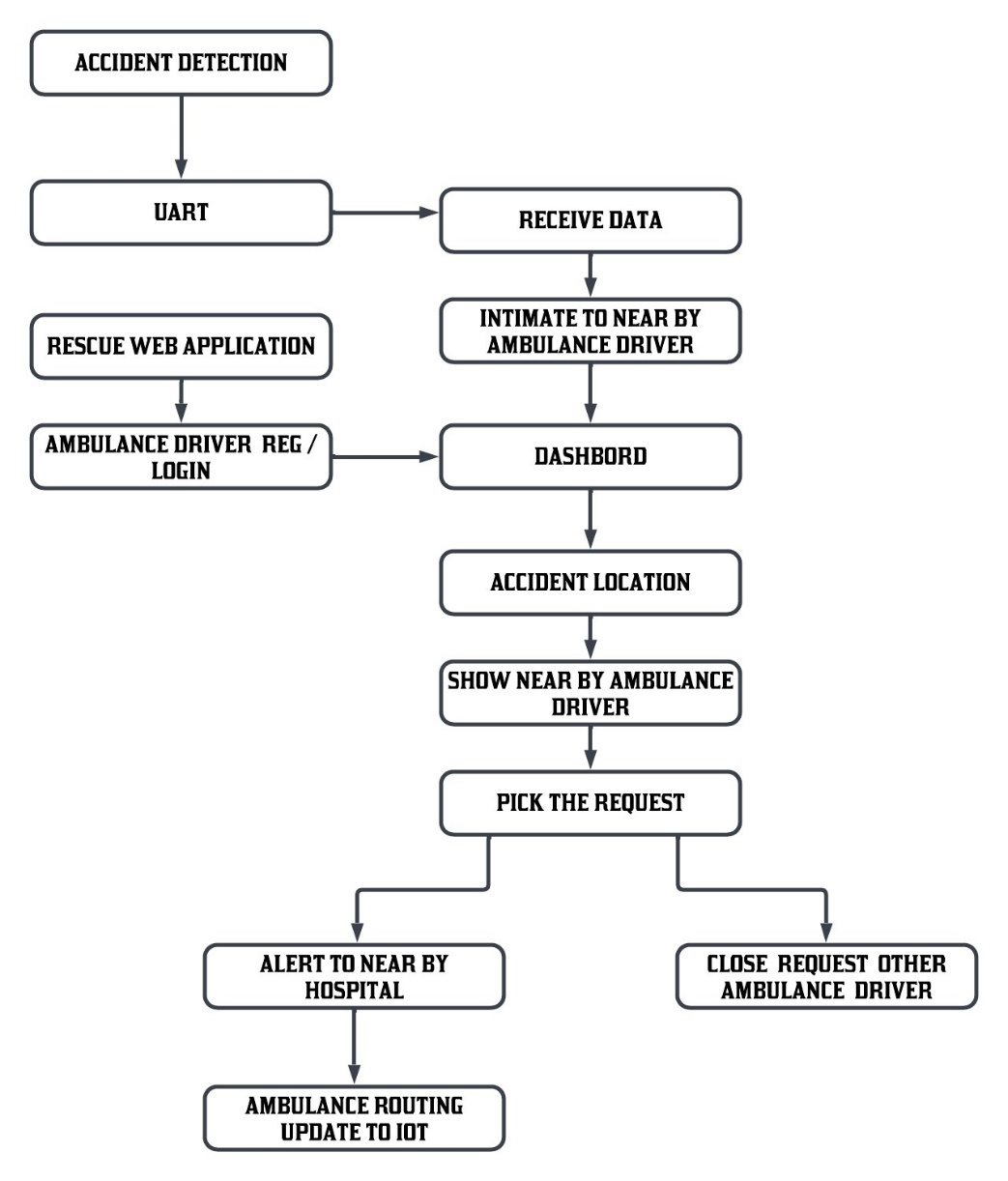


**LEVEL: 3**



**LEVEL: 4**



**OVERALL DIAGRAM:**

**4.8 PROPOSED SYSTEM ALGORITHM:**

**SOFTWARE REQUIREMENT SPECIFICATION**

**INTRODUCTION TO PYTHON**

Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

* Web Development
* Software Development
* Game Development
* AI & ML
* Data Analytics

This list can go on as we go but why python is so much popular let’s see it in the next topic.

**WHY PYTHON PROGRAMMING?**

You guys might have a question in mind that, why python? why not another programming language? So let me explain:

Every Programming language serves some purpose or use-case according to a domain. for eg, JavaScript is the most popular language amongst web developers as it gives the developer the power to handle applications via different frameworks like react, vue, angular which are used to build beautiful User Interfaces. Similarly, they have pros and cons at the same time. so if we consider python, it is general-purpose which means it is widely used in every domain the reason is it’s very simple to understand, scalable because of which the speed of development is so fast. Now you get the idea why besides learning python it doesn’t require any programming background so that’s why it’s popular amongst developers as well. Python has simpler syntax similar to the English language and also the syntax allows developers to write programs with fewer lines of code. Since it is open-source there are many libraries available that make developers’ jobs easy ultimately results in high productivity. They can easily focus on business logic and Its demanding skills in the digital era where information is available in large data sets.

**HOW DO WE GET STARTED?**

Now in the era of the digital world, there is a lot of information available on the internet that might confuse us believe me. what we can do is follow the documentation which is a good start point. Once we are familiar with concepts or terminology, we can dive deeper into this.

Following are references where we can start our journey:

Official Website: https://www.python.org/

Udemy Course: https://www.udemy.com/course/python-the-complete-python-developercourse/

YouTube: https://www.youtube.com/watch?v=\_uQrJ0TkZlc

CodeAcademy: https://www.codecademy.com/catalog/language/python

I hope now you guys are excited to get started right so you might be wondering where we can start coding right so there are a lot of options available in markets. we can use any IDE we are comfortable with but for those who are new to the programming world I am listing some of IDE’s below for python:

1. Visual Studio: https://visualstudio.microsoft.com/
2. PyCharm: https://www.jetbrains.com/pycharm/
3. Spyder: https://www.spyder-ide.org/
4. Atom: https://atom.io/
5. Google Collab: https://research.google.com/colaboratory/

**Real-World Examples:**

**1) NASA (National Aeronautics and Space Agency):** One of Nasa’s Shuttle Support Contractors, United Space Alliance developed a Workflow Automation System (WAS) which is fast. Internal Resources Within critical project stated that:

“Python allows us to tackle the complexity of programs like the WAS without getting bogged down in the language”.

Nasa also published a website (https://code.nasa.gov/) where there are 400 open-source projects which use python.

**2) Netflix**: There are various projects in Netflix which use python as follow:

* Central Alert Gateway
* Chaos Gorilla
* Security Monkey
* Chronos

Amongst all projects, regional failover is the project they have as the system decreases outage time from 45 minutes to 7 minutes with no additional cost.

**3)** **Instagram:** Instagram also uses python extensively. They have built a photo-sharing social platform using Django which is a web framework for python. Also, they are able to successfully upgrade their framework without any technical challenges.

**Applications of Python Programming:**

1. **Web Development**: Python offers different frameworks for web development like Django, Pyramid, Flask. This framework is known for security, flexibility, scalability.
2. **Game Development:** PySoy and PyGame are two python libraries that are used for game development
3. **Artificial Intelligence and Machine Learning:** There is a large number of open-source libraries which can be used while developing AI/ML applications.
4. **Desktop GUI:** Desktop GUI offers many toolkits and frameworks using which we can build desktop applications.PyQt, PyGtk, PyGUI are some of the GUI frameworks.

**How to Become Better Programmer:**

The last but most important thing is how you get better at what programming you choose is practice practice practice. Practical knowledge only acquired by playing with things so you will get more exposure to real-world scenarios. Consistency is more important than anything because if you practice it for some days and then you did nothing then when you start again it will be difficult to practice consistently. So I request you guys to learn by doing projects so it will help you understand how things get done and important thing is to have fun at the same time.

**Approach to be followed to master Python:**

**“Beginning is the end and end is the beginning”.** I know what you are thinking about. It is basically a famous quote from a web series named “Dark”. Now how it relates to Python programming?

If you researched on google, youtube, or any development communities out there, you will find that people explained how you can master programming in let’s say some “x” number of days and like that.

Well, the reality is like the logo of infinity which we can see above. In the programming realm, there is no such thing as mastery. It’s simply a trial and error process. For example. Yesterday I was writing some code where I was trying to print a value of a variable before declaring it inside a function. There I had seen a new error named “**UnboundLocalErrorException** “.

So the important thing to keep in mind is that programming is a surprising realm. Throughout your entire career, you will be seeing new errors and exceptions. Just remember the quote – **“Practise makes a man perfect”.**

Now here is the main part. What approach to follow in order to master Python Programming?

Well, here it is:

**Step-1: Start with a “Hello World” Program**

If you happened to learn some programming languages, then I am sure you are aware of what

I am talking about. The “Hello World” program is like a tradition in the developer community. If you want to master any programming language, this should be the very first line of code we should be seeking for.

**Simple Hello World Program in Python:**

print ("Hello World")

**Step-2: Start learning about variables**

Now once we have mastered the “Hello World” program in Python, the next step is to master variables in python. Variables are like containers that are used to store values.

**Variables in Python:**

my\_var = 100

As you can see here, we have created a variable named “my\_var” to assign a value 100 to the same.

**Step-3: Start learning about Data Types and Data Structures**

The next outpost is to learn about data types. Here I have seen that there is a lot of confusion between data types and data structures. The important thing to keep in mind here is that data types represent the type of data. For example. in Python, we have something like int, string, float, etc. Those are called data types as they indicate the type of data we are dealing with.

While data structures are responsible for deciding how to store this data in a computer’s memory.

**String data type in Python:**

my\_str = "ABCD"

As you can see here, we have assigned a value “ABCD” to a variable my\_str. This is basically a string data type in Python.

**Data Structure in Python:**

my\_dict= {1:100,2:200,3:300}

This is known as a dictionary data structure in Python.

Again, this is just the tip of the iceberg. There are lots of data types and data structures in Python. To give a basic idea about data structures in Python, here is the complete list:

1.Lists

2.Dictionary

3.Sets

4.Tuples

5.Frozenset

**Step-4: Start learning about conditionals and loops**

In any programming language, conditionals and loops are considered one of the backbone.

Python is no exception for that as well. This is one of the most important concepts that we need to master.

**IF-ELIF-ELSE conditionals:**

if(x < 10):

print("x is less than 10") elif(x > 10):

print("x is greater than 10") else:

print("Do nothing")

As you can see in the above example, we have created what is known as the if-elif else ladder **For loop:**

for i in "Python":

print(i)

The above code is basically an example of for loop in python.

**PRO Tip:**

Once you start programming with Python, you will be seeing that if we missed any white spacing in python then python will start giving some errors. This is known as Indentation in python. Python is very strict with indentation. Python is created with a mindset to help everyone become a neat programmer. This indentation scheme in python is introduced in one of python’s early PEP (Python Enhancement Proposal).

## THE PYTHON STANDARD LIBRARY

While [The Python Language Reference](https://docs.python.org/3/reference/index.html#reference-index) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions. Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs. The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components. In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index.](https://pypi.org/)

What Is a Python Package?

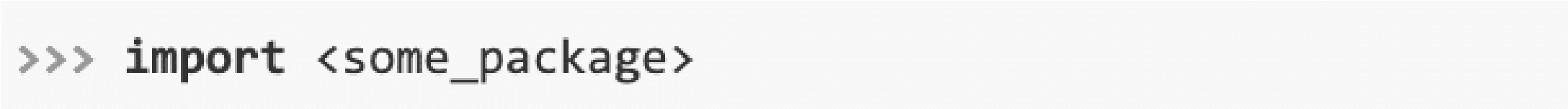
To understand Python packages, we’ll briefly look at scripts and modules. A “script” is something you execute in the shell to accomplish a defined task. To write a script, you’d type your code into your favourite [text editor](https://hackr.io/blog/best-python-ide) and save it with the .py extension. You can then use the python command in a terminal to execute your script. A module on the other hand is a Python program that you import, either in [interactive mode](https://docs.python.org/3/tutorial/interpreter.html#interactive-mode) or into your other programs. “Module” is really an umbrella term for reusable code. A Python package usually consists of several modules. Physically, a package is a folder containing modules and maybe other folders that themselves may contain more folders and modules. Conceptually, it’s a namespace. This simply means that a package’s modules are bound together by a package name, by which they may be referenced. Circling back to our earlier definition of a module as reusable, importable code, we note that every package is a module — but not every module is a package. A package folder usually contains one file named \_\_init\_\_.py that basically tells Python: “Hey, this directory is a package!” The init file may be empty, or it may contain code to be executed upon package initialization. You’ve probably come across the term “library” as well. For Python, a library isn’t as clearly defined as a package or a module, but a good rule of thumb is that whenever a package has been published, it may be referred to as a library.

**HOW TO USE A PYTHON PACKAGE**

We’ve mentioned namespaces, publishing packages and importing modules. If any of these terms or concepts aren’t entirely clear to you, we’ve got you! In this section, we’ll cover everything you’ll need to really grasp the pipeline of using Python packages in your code. Importing a Python Package

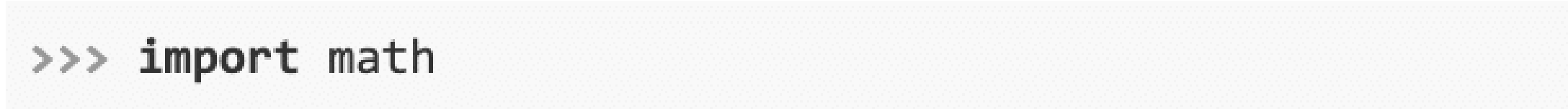
We’ll import a package using the **import** statement:

Let’s assume that we haven’t yet installed any packages. Python comes with a big collection of pre-installed packages known as the Python Standard Library. It includes tools for a range of use cases, such as text processing and doing math. Let’s import the latter:

You might think of an import statement as a search trigger for a module. Searches are strictly organized: At first, Python looks for a module in the cache, then in the standard library and finally in a list of paths. This list may be accessed after importing sys (another standard library module). 

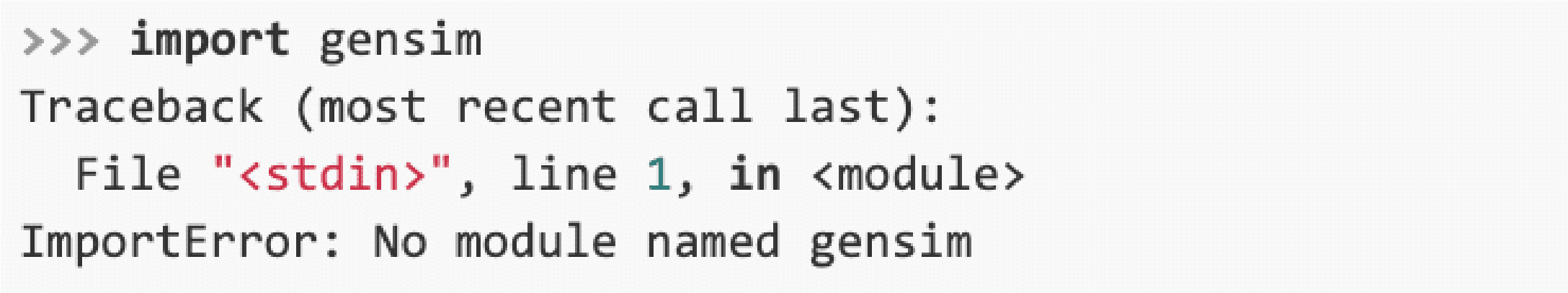
The sys.path command returns all the directories in which Python will try to find a package.

It may happen that you’ve downloaded a package but when you try importing it, you get an error:

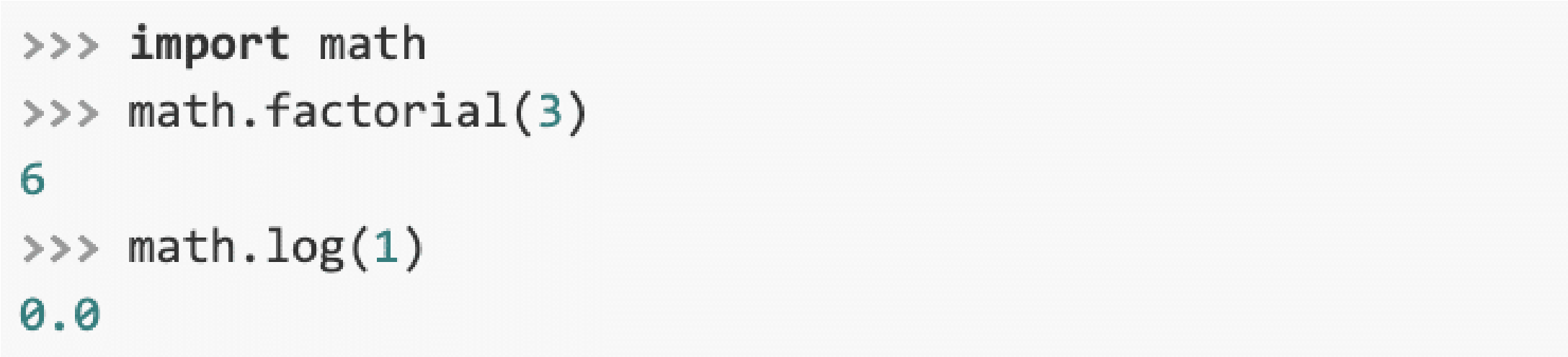
In such cases, check whether your imported package has been placed in one of Python’s search paths. If it hasn’t, you can always expand your list of search paths:

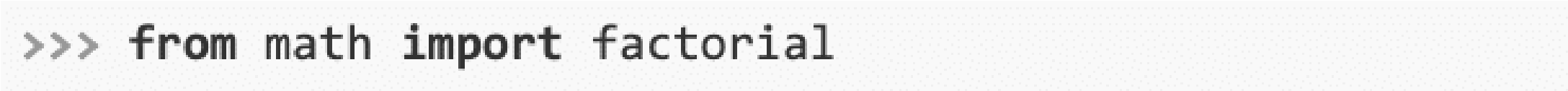
At that point, the interpreter will have more than one more location to look for packages after receiving an **import** statement.

Namespaces and Aliasing

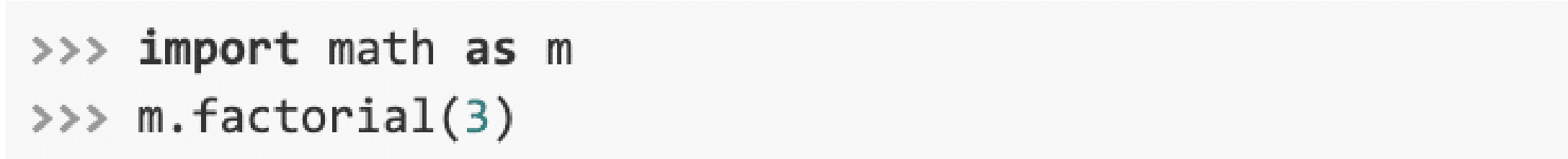
When we had imported the math module, we initialized the math namespace. This means that we can now refer to functions and classes from the math module by way of “dot notation”:

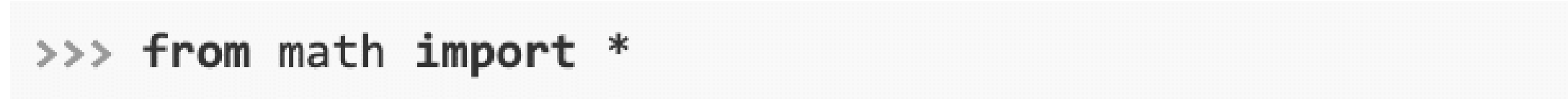
Assume that we were only interested in our math module’s factorial function, and that we’re also tired of using dot notation. In that case, we can proceed as follows:

If you’d like to import multiple resources from the same source, you can simply comma separate them in the import statement:

There is, however, always a small risk that your variables will clash with 

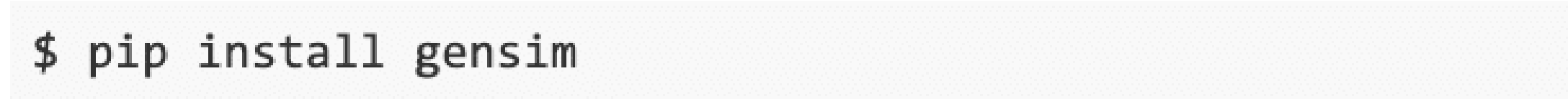
other variables in your namespace. What if one of the variables in your code was named log, too?

It would overwrite the log function, causing bugs. To avoid that, it’s better to import the package as we did before. If you want to save typing time, you can alias your package to give it a shorter name:

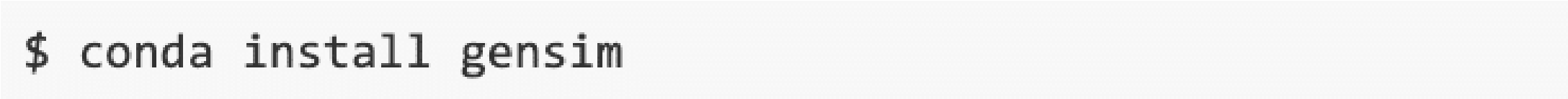
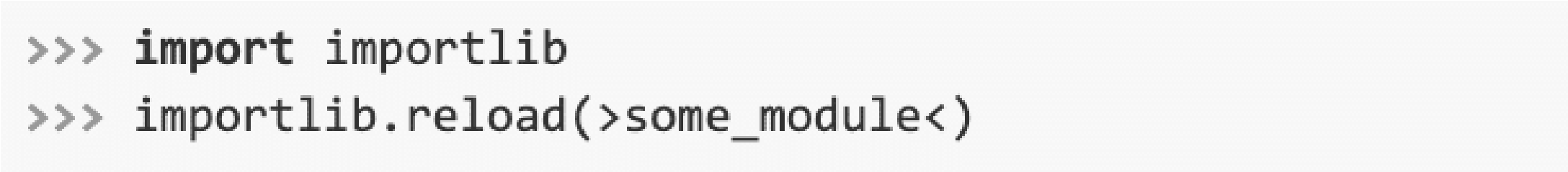
Aliasing is a pretty common technique. Some packages have commonly used aliases: For instance, the numerical computation library NumPy is almost always imported as “np.” Another option is to import all a module’s resources into your namespace:

However, this method poses serious risk since you usually don’t know all the names contained in a package, increasing the likelihood of your variables being overwritten. It’s for this reason that most seasoned Python programmers will discourage use of the wildcard

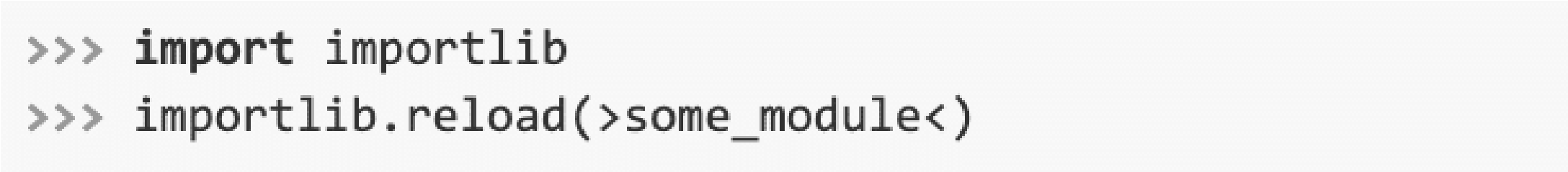
\* in imports. Also, as the [Zen of Python](https://www.python.org/dev/peps/pep-0020/) states, “namespaces are one honking great idea!”

How to Install a Python Package

How about packages that are not part of the standard library? The official repository for finding and downloading such third-party packages is the Python Package Index, usually referred to simply as [PyPI.](https://pypi.org/) To install packages from PyPI, use the package installer [pip:](https://packaging.python.org/tutorials/installing-packages/)

pip can install Python packages from any source, not just PyPI. If you installed Python using [Anaconda](https://www.anaconda.com/) or [Miniconda,](https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html#anaconda-or-miniconda) you can also use the conda command to install Python packages.

While conda is very easy to use, it’s not as versatile as pip. So if you cannot install a package using conda, you can always try pip instead. Reloading a Module

If you’re programming in interactive mode, and you change a module’s script, these changes won’t be imported, even if you issue another import statement. In such case, you’ll want to use the reload() function from the importlib library:

**How to Create Your Own Python Package?**

Packaging your code for further use doesn’t necessarily mean you’ll want it published to PyPI. Maybe you just want to share it with a friend, or reuse it yourself. Whatever your aim, there are several files that you should include in your project. We’ve already mentioned the \_\_init\_\_.py file.

Another important file is setup.py. Using the setuptools package, this file provides detailed information about your project and lists all dependencies — packages required by your code to run properly.

Publishing to PyPI is beyond the scope of this introductory tutorial. But if you do have a package for distribution, your project should include two more files: a README.md written in Markdown, and a license. Check out the official Python Packaging User Guide [(PyPUG)](https://packaging.python.org/tutorials/packaging-projects/) if you want to know more.

**INSTALLING PACKAGES**

This section covers the basics of how to install Python [packages.](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) It’s important to note that the term “package” in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a [distribution)](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package). It does not to refer to the kind of [package](https://packaging.python.org/en/latest/glossary/#term-Import-Package) that you import in your Python source code (i.e. a container of modules). It is common in the Python community to refer to a [distribution](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) using the term “package”. Using the term “distribution” is often not preferred, because it can easily be confused with a Linux distribution, or Python itself.

[**Requirements for Installing Packages**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id11)

This section describes the steps to follow before installing other Python packages.

[**Ensure you can run Python from the command line**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id12)

Before you go any further, make sure you have Python and that the expected version is available from your command line. You can check this by running:

 Unix/macOS

python3 --version

 Windows

You should get some output like Python 3.6.3. If you do not have Python, please install the latest 3.x version from [python.org](https://www.python.org/) or refer to the [Installing Python](https://docs.python-guide.org/starting/installation/#installation) section of the Hitchhiker’s Guide to Python.

**Note**

If you’re a newcomer and you get an error like this:

**>>>** python --version

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError

:

name 'python' is not defined

It’s because this command and other suggested commands in this tutorial are intended to be

run in a

*shell*

(Also called a

*terminal*

or

*Console)*

See the Python for Beginners

[getting started](https://opentechschool.github.io/python-beginners/en/getting_started.html#what-is-python-exactly)

[tutoria](https://opentechschool.github.io/python-beginners/en/getting_started.html#what-is-python-exactly)l

for an introduction to using your operating system’s shell and interacting with

Python.

**Note**

|  |
| --- |
| If you’re using an enhanced shell like IPython or the Jupyter notebook, you can run system commands like those in this tutorial by prefacing them with a ! character: |

In [1]: import sys

!{sys.executable} --version

Python 3.6.3

It’s recommended to write {sys.executable} rather than plain python in order to

ensure that commands are run in the Python installation matching the

currently running notebook (which may not be the same Python installation

that the python command refers to).

## Note

Due to the way most Linux distributions are handling the Python 3 migration, Linux

users using the system Python without creating a virtual environment first should replace

the python command in this tutorial with python3 and the python -m pip command

with python3 -m pip --user. Do *not* run any of the commands in this tutorial with sudo: if

you get a permissions error, come back to the section on creating virtual environments,

set one up, and then continue with the tutorial as written.

[**Ensure you can run pip from the command line**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id13)

Additionally, you’ll need to make sure you have [pip](https://packaging.python.org/en/latest/key_projects/#pip) available. You can check this by running:

 Unix/macOS

python3 -m pip --version

 Windows

If you installed Python from source, with an installer from [python.org,](https://www.python.org/) or via [Homebrew](https://brew.sh/) you should already have pip. If you’re on Linux and installed using your OS package manager, you may have to install pip separately, see [Installing pip/setuptools/wheel with Linux Package Managers.](https://packaging.python.org/en/latest/guides/installing-using-linux-tools/)

If pip isn’t already installed, then first try to bootstrap it from the standard library:

 Unix/macOS

python3 -m ensurepip --default-pip

 Windows

If that still doesn’t allow you to run python -m pip:

* Securely Download [get-pip.py](https://bootstrap.pypa.io/get-pip.py) [1](https://packaging.python.org/en/latest/tutorials/installing-packages/#id7)
* Run python get-pip.py. [2](https://packaging.python.org/en/latest/tutorials/installing-packages/#id8) This will install or upgrade pip. Additionally, it will install [setuptools](https://packaging.python.org/en/latest/key_projects/#setuptools) and [wheel](https://packaging.python.org/en/latest/key_projects/#wheel) if they’re not installed already.

**Warning**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Be cautious if you’re using a Python install that’s managed by your operating system or another package manager. get-pip.py does not coordinate with those tools, and may | | |
| leave your system in an inconsistent state. You can use | | |  | | --- | | P ython get-pip.py -- | |
| -installed |
| |  | | --- | | prefix=/usr/local/ | | |  | | --- | | /usr/local |   to install in which is designed for locally |
| software. |
| [**Ensure pip, setuptools, and wheel are up to date**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id14) | | | |

While pip alone is sufficient to install from pre-built binary archives, up to date copies of the setuptools and wheel projects are useful to ensure you can also install from source archives:

 Unix/macOS

python3 -m pip install --upgrade pip setuptools wheel

 Windows

[**Optionally, create a virtual environment**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id15)

See [section below](https://packaging.python.org/en/latest/tutorials/installing-packages/#creating-and-using-virtual-environments) for details, but here’s the basic [venv](https://docs.python.org/3/library/venv.html) [3](https://packaging.python.org/en/latest/tutorials/installing-packages/#id9) command to use

on a typical Linux system:

 Unix/macOS

python3 -m venv tutorial\_env source tutorial\_env/bin/activate

 Windows

This will create a new virtual environment in the tutorial\_env subdirectory, and configure the current shell to use it as the default python environment.

[**Creating Virtual Environments**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id16)

Python “Virtual Environments” allow Python [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see [Installing stand alone command line tools.](https://packaging.python.org/en/latest/guides/installing-stand-alone-command-line-tools/)

Imagine you have an application that needs version 1 of LibFoo, but another application requires version 2. How can you use both these applications? If you install everything into /usr/lib/python3.6/site-packages (or whatever your platform’s standard location is), it’s easy to end up in a situation where you unintentionally upgrade an application that shouldn’t be upgraded.

Or more generally, what if you want to install an application and leave it be? If an application works, any change in its libraries or the versions of those libraries can break the application.

Also, what if you can’t install [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) into the global site-packages directory? For instance, on a shared host.

In all these cases, virtual environments can help you. They have their own installation directories and they don’t share libraries with other virtual environments. Currently, there are two common tools for creating Python virtual environments:

* [venv](https://docs.python.org/3/library/venv.html) is available by default in Python 3.3 and later, and installs [pip](https://packaging.python.org/en/latest/key_projects/#pip) and [setuptools](https://packaging.python.org/en/latest/key_projects/#setuptools) into created virtual environments in Python 3.4 and later.
* [virtualenv](https://packaging.python.org/en/latest/key_projects/#virtualenv) needs to be installed separately, but supports Python 2.7+ and Python 3.3+, and [pip,](https://packaging.python.org/en/latest/key_projects/#pip) [setuptools](https://packaging.python.org/en/latest/key_projects/#setuptools) and [wheel](https://packaging.python.org/en/latest/key_projects/#wheel) are always installed into created virtual environments by default (regardless of Python version).

The basic usage is like so:

Using [venv:](https://docs.python.org/3/library/venv.html)

 Unix/macOS

python3 -m venv <DIR> source <DIR>/bin/activate

 Windows

Using [virtualenv:](https://packaging.python.org/en/latest/key_projects/#virtualenv)

 Unix/macOS

python3 -m virtualenv <DIR> source <DIR>/bin/activate

 Windows

For more information, see the [venv](https://docs.python.org/3/library/venv.html) docs or the [virtualenv](https://virtualenv.pypa.io/en/stable/index.html) docs.

The use of **source** under Unix shells ensures that the virtual environment’s variables are set within the current shell, and not in a subprocess (which then disappears, having no useful effect).

In both of the above cases, Windows users should \_not\_ use the **source** command, but should rather run the **activate** script directly.

<DIR>\Scripts\activate

Managing multiple virtual environments directly can become tedious, so the [dependency management tutorial](https://packaging.python.org/en/latest/tutorials/managing-dependencies/#managing-dependencies) introduces a higher level tool, [Pipenv,](https://packaging.python.org/en/latest/key_projects/#pipenv) that automatically manages a separate virtual environment for each project and application that you work on. [**Use pip for Installing**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id17)  [pip](https://packaging.python.org/en/latest/key_projects/#pip) is the recommended installer. Below, we’ll cover the most common usage scenarios. For more detail, see the [pip docs,](https://pip.pypa.io/en/latest/) which includes a complete [Reference Guide.](https://pip.pypa.io/en/latest/cli/)

[**Installing from PyPI**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id18)

The most common usage of [pip](https://packaging.python.org/en/latest/key_projects/#pip) is to install from the [Python Package Index](https://packaging.python.org/en/latest/glossary/#term-Python-Package-Index-PyPI) using a [requirement specifier.](https://packaging.python.org/en/latest/glossary/#term-Requirement-Specifier) Generally speaking, a requirement specifier is composed of a project name followed by an optional [version specifier.](https://packaging.python.org/en/latest/glossary/#term-Version-Specifier) [**PEP 440**](https://www.python.org/dev/peps/pep-0440) contains a [**full specification**](https://www.python.org/dev/peps/pep-0440#version-specifiers) of the currently supported specifiers. Below are some examples.

To install the latest version of “SomeProject”:

 Unix/macOS

python3 -m pip install "SomeProject"

 Windows

To install a specific version:

 Unix/macOS

python3 -m pip install "SomeProject==1.4"

 Windows

To install greater than or equal to one version and less than another:

 Unix/macOS

python3 -m pip install "SomeProject>=1,<2"

 Windows

To install a version that’s [**“compatible”**](https://www.python.org/dev/peps/pep-0440#compatible-release) with a certain version: [4](https://packaging.python.org/en/latest/tutorials/installing-packages/#id10)

 Unix/macOS

python3 -m pip install "SomeProject~=1.4.2"

 Windows

In this case, this means to install any version “==1.4.\*” version that’s also “>=1.4.2”. [**Source Distributions vs Wheels**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id19) [pip](https://packaging.python.org/en/latest/key_projects/#pip) can install from either [Source Distributions (sdist)](https://packaging.python.org/en/latest/glossary/#term-Source-Distribution-or-sdist) or [Wheels,](https://packaging.python.org/en/latest/glossary/#term-Wheel) but if both are present on PyPI, pip will prefer a compatible [wheel.](https://packaging.python.org/en/latest/glossary/#term-Wheel) You can override pip`s default behavior by e.g. using its [–no-binary](https://pip.pypa.io/en/latest/cli/pip_install/#install-no-binary) option. [Wheels](https://packaging.python.org/en/latest/glossary/#term-Wheel) are a pre-built [distribution](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) format that provides faster installation compared to [Source Distributions (sdist),](https://packaging.python.org/en/latest/glossary/#term-Source-Distribution-or-sdist) especially when a project contains compiled extensions. If [pip](https://packaging.python.org/en/latest/key_projects/#pip) does not find a wheel to install, it will locally build a wheel and cache it for future installs, instead of rebuilding the source distribution in the future.

[**Upgrading packages**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id20)

Upgrade an already installed SomeProject to the latest from PyPI.

 Unix/macOS

python3 -m pip install --upgrade SomeProject

 Windows

[**Installing to the User Site**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id21)

To install [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) that are isolated to the current user, use the --user flag:

 Unix/macOS

python3 -m pip install --user SomeProject

 Windows

For more information see the [User Installs](https://pip.pypa.io/en/latest/user_guide/#user-installs) section from the pip docs.

Note that the --user flag has no effect when inside a virtual environment - all installation commands will affect the virtual environment.

If SomeProject defines any command-line scripts or console entry points, --user will cause them to be installed inside the [user base’](https://docs.python.org/3/library/site.html#site.USER_BASE)s binary directory, which may or may not already be present in your shell’s PATH. (Starting in version 10, pip displays a warning when installing any scripts to a directory outside PATH.) If the scripts are not available in your shell after installation, you’ll need to add the directory to your PATH:

* On Linux and macOS you can find the user base binary directory by running python m site --user-base and adding bin to the end. For example, this will typically print ~/.local (with ~ expanded to the absolute path to your home directory) so you’ll need to add ~/.local/bin to your PATH. You can set your PATH permanently by [modifying ~/.profile.](https://stackoverflow.com/a/14638025)
* On Windows you can find the user base binary directory by running py -m site --usersite and replacing site-packages with Scripts. For example, this could return C:\Users\Username\AppData\Roaming\Python36\site-packages so you would need to set your PATH to include C:\Users\Username\AppData\Roaming\Python36\Scripts. You can set your user PATH permanently in the [Control Panel.](https://docs.microsoft.com/en-us/windows/win32/shell/user-environment-variables?redirectedfrom=MSDN) You may need to log out for the PATH changes to take effect. [**Requirements files**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id22) Install a list of requirements specified in a [Requirements File.](https://pip.pypa.io/en/latest/user_guide/#requirements-files)

 Unix/macOS

python3 -m pip install -r requirements.txt

 Windows

[**Installing from VCS**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id23)

Install a project from VCS in “editable” mode. For a full breakdown of the syntax, see pip’s section on [VCS Support.](https://pip.pypa.io/en/latest/cli/pip_install/#vcs-support)

 Unix/macOS

python3 -m pip install -e git+https://git.repo/some\_pkg.git#egg=SomeProject # from git python3 -m pip install -e hg+https://hg.repo/some\_pkg#egg=SomeProject # from mercurial python3 -m pip install -e svn+svn://svn.repo/some\_pkg/trunk/#egg=SomeProject # from svn

python3 -m pip install -e git+https://git.repo/some\_pkg.git@feature#egg=SomeProject # from a branch

 Windows

[**Installing from other Indexes**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id24)

Install from an alternate index

 Unix/macOS

python3 -m pip install --index-url http://my.package.repo/simple/ SomeProject

 Windows

Search an additional index during install, in addition to [PyPI](https://packaging.python.org/en/latest/glossary/#term-Python-Package-Index-PyPI)

 Unix/macOS

python3 -m pip install --extra-index-url http://my.package.repo/simple SomeProject

 Windows

[**Installing from a local src tree**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id25)

Installing from local src in [Development Mode,](https://setuptools.pypa.io/en/latest/userguide/development_mode.html) i.e. in such a way that the project appears to be installed, but yet is still editable from the src tree.

 Unix/macOS

python3 -m pip install -e <path>

 Windows

You can also install normally from src

 Unix/macOS

python3 -m pip install <path>

 Windows

[**Installing from local archives**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id26)

Install a particular source archive file.

 Unix/macOS

python3 -m pip install ./downloads/SomeProject-1.0.4.tar.gz

 Windows

Install from a local directory containing archives (and don’t check [PyPI)](https://packaging.python.org/en/latest/glossary/#term-Python-Package-Index-PyPI)

 Unix/macOS

python3 -m pip install --no-index --find-links=file:///local/dir/ SomeProject python3 -m pip install --no-index --find-links=/local/dir/ SomeProject python3 -m pip install --no-index --find-links=relative/dir/ SomeProject

 Windows

[**Installing from other sources**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id27)

To install from other data sources (for example Amazon S3 storage) you can create a helper application that presents the data in a [**PEP 503**](https://www.python.org/dev/peps/pep-0503) compliant index format, and use the --extraindex-url flag to direct pip to use that index.

./s3helper --port=7777 python -m pip install --extra-index-url http://localhost:7777 SomeProject

[**Installing Prereleases**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id28)

Find pre-release and development versions, in addition to stable versions. By default, pip only finds stable versions.

 Unix/macOS

python3 -m pip install --pre SomeProject

 Windows

[**Installing Setuptools”**](https://packaging.python.org/en/latest/tutorials/installing-packages/#id29)

Install [setuptools extras.](https://setuptools.readthedocs.io/en/latest/userguide/dependency_management.html#optional-dependencies)

* Unix/macOS

python3 -m pip install SomePackage[PDF] python3 -m pip install SomePackage[PDF]==3.0 python3 -m pip install -e .[PDF] # editable project in current directory

**4.9 SOFTWARE TESTING**

## UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

1. **FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

* Valid Input: identified classes of valid input must be accepted.
* Invalid Input: identified classes of invalid input must be rejected.
* Functions: identified functions must be exercised.
* Output: identified classes of application outputs must be exercised.

1. **SYSTEMS PROCEDURES: INTERFACING SYSTEMS OR PROCEDURES MUST BE INVOKED.**

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

## WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

## BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document.

## FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system.

**Chapter 5**

**RESULTS**

**AND DISCUSSION**

**CHAPTER 5**

**RESULTS AND DISCUSSION**

The Accident Detection and Alert System was successfully designed and implemented to detect road accidents in real time and send immediate alerts to emergency contacts or authorities. The system utilized sensors such as accelerometers, gyroscopes, and vibration sensors to detect sudden impacts and collisions. Upon detecting an accident, the system automatically transmitted an alert message, including the GPS location, via a GSM/GPRS module to ensure a quick emergency response. The system was tested in various conditions, demonstrating high accuracy in accident detection and a fast response time of under 10 seconds for alert generation. The reliability and efficiency of the system were validated through multiple trials in different environments, including urban and highway settings. Overall, the project successfully achieved its objective of enhancing road safety by providing a real-time, automated accident detection and alert mechanism. Future improvements may include integrating AI-based accident prediction, voice-assisted emergency responses, and improved network connectivity in remote areas to further enhance the system’s effectiveness.

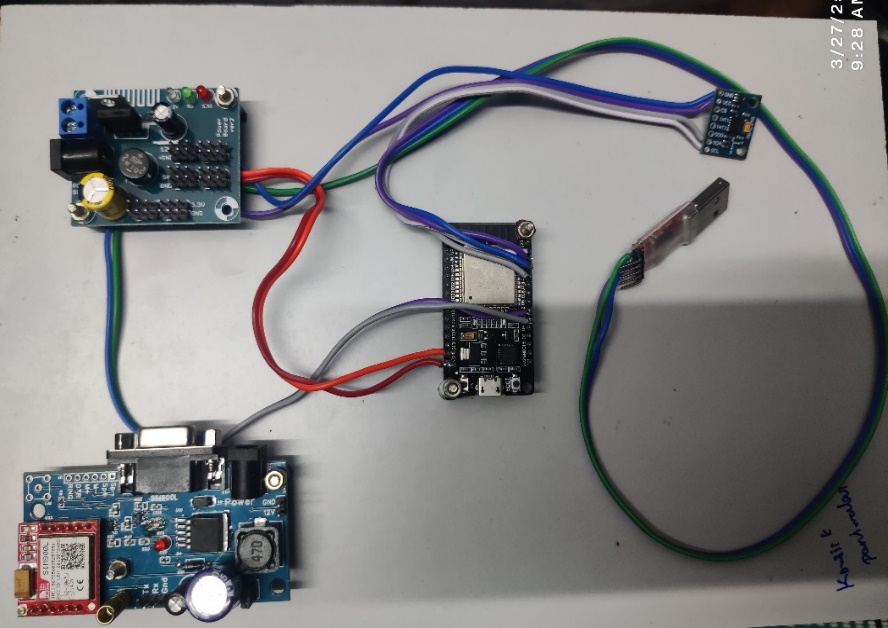


Fig 7. Final Prototype

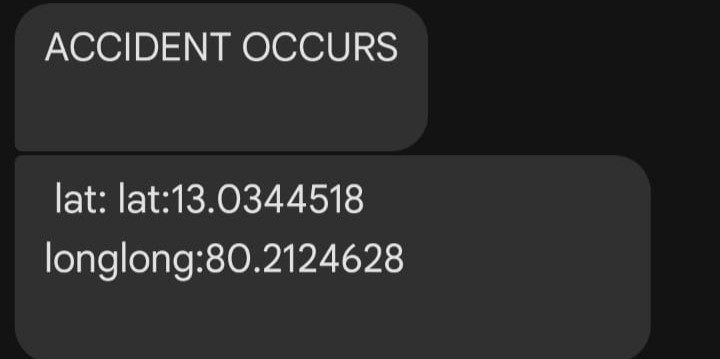


Fig 8. Notification message

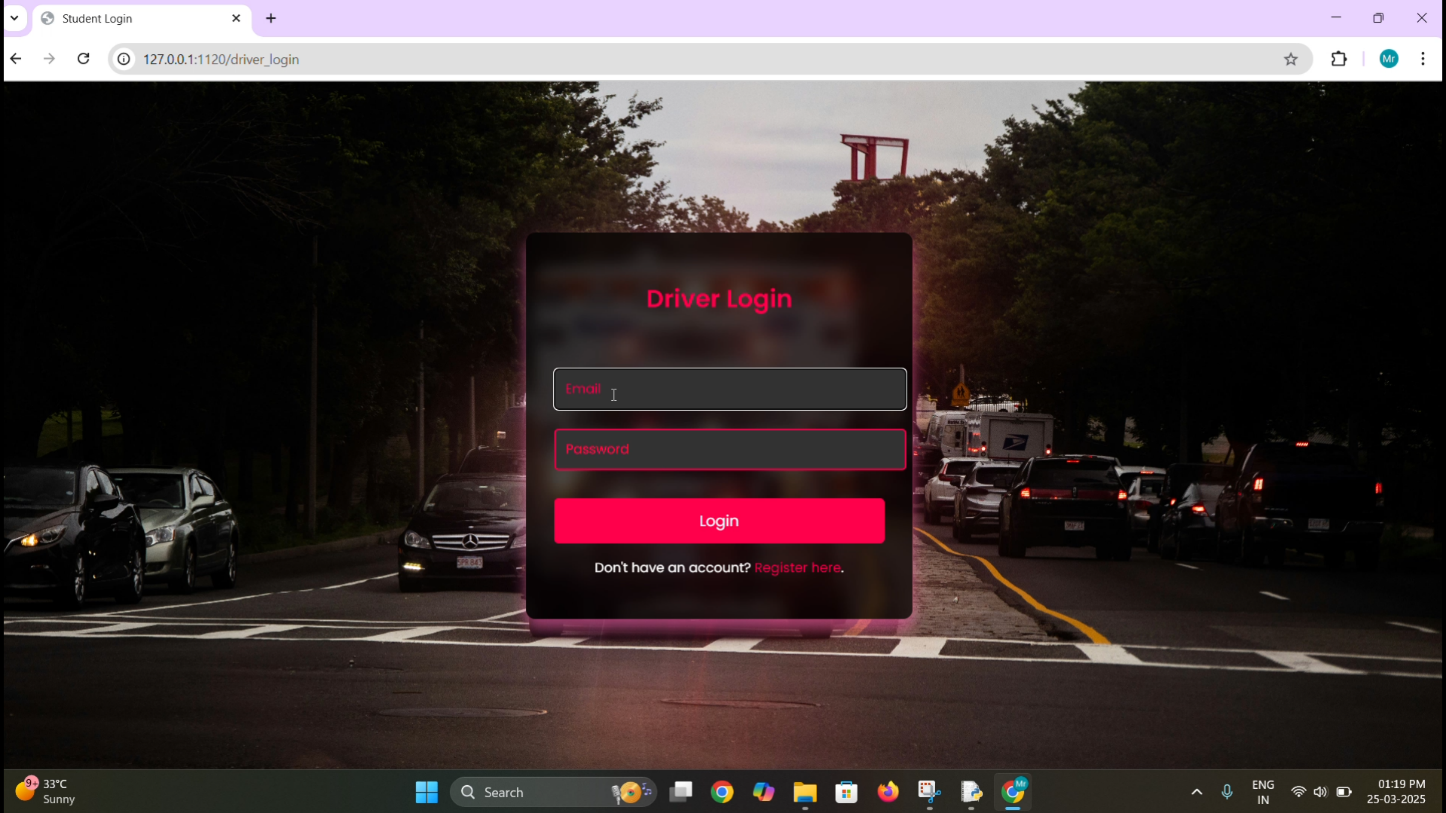


Fig 9. Web Page Application

Web

Fig 10: Experimental Graph



Fig 11: Experiment results. (a) comparison of accident detected; (b) accuracy percentage of experiments; (c) false reporting of experiments; (d) parameter-based comparison

overall result of this project is an application that provides help to people who require help but can’t ask for it. With the help of the application, their request for help is sent at the time of the accident with their location which helps emergency services provide support as early and effective as possible. All this is done with only the sensors available at low cost.

Chapter 6

CONCLUSION

\

CHAPTER 6

CONCLUSION

The **Real-Time Accident Detection and Emergency Response System** is a transformative innovation designed to enhance road safety by significantly reducing emergency response times. With the integration of **advanced sensors, IoT, GSM communication, and GPS tracking**, the system ensures that accidents are detected instantly, and emergency services are alerted in real time. Unlike traditional accident detection systems that rely on manual reporting, which can be delayed due to human error or communication gaps, this system **automates the entire emergency response process**, ensuring a quicker and more reliable dispatch of ambulances to accident sites. This rapid intervention can make a crucial difference in life-threatening situations, increasing the survival rate of accident victims. A major advantage of this system is its ability to **eliminate redundant ambulance dispatches**, ensuring that only the nearest available ambulance is assigned to the emergency. This prevents resource wastage and avoids confusion among emergency responders. Additionally, **real-time GPS tracking and route optimization** help ambulances navigate efficiently, minimizing delays caused by heavy traffic or roadblocks. The incorporation of **IoT-based monitoring** allows continuous tracking and logging of accident data, which can be analysed to identify accident-prone areas, improve traffic management, and develop more effective road safety policies. Looking ahead, this system has immense potential for **further enhancement and integration with emerging technologies**. Implementing **AI-based accident prediction models** can allow the system to anticipate potential accident hotspots and take preventive measures. **Vehicle-to-Everything (V2X) communication** can improve coordination between emergency vehicles, traffic management systems, and hospitals, ensuring seamless emergency handling. Cloud-based **data analytics and machine learning** can refine accident detection accuracy and optimize ambulance dispatching based on historical data. In conclusion, this project serves as a **pioneering step toward revolutionizing emergency response mechanisms**. By leveraging cutting-edge technologies, the system enhances public safety, reduces fatalities, and optimizes the use of emergency resources. With continuous research, development, and widespread adoption, this system has the potential to become a **standard feature in smart transportation infrastructure**, making roads safer and emergency medical responses more efficient worldwide.

## REFERENCES

[1]. S. Li, "Fall Detection with Wrist-Worn Watch by Observations in Statistics of Acceleration," in *IEEE Access*, vol. 11, pp. 19567-19578, 2023, doi: 10.1109/ACCESS.2023.3249191.

[2]. A. Jarndal, H. Tawfik, A. I. Siam, I. Alsyouf and A. Cheaitou, "A Real-Time Vision Transformers-Based System for Enhanced Driver Drowsiness Detection and Vehicle Safety," in *IEEE Access*, vol. 13, pp. 1790-1803, 2025, doi: 10.1109/ACCESS.2024.3522111.

[3]. U. Alvi, M. A. K. Khattak, B. Shabir, A. W. Malik and S. R. Muhammad, "A Comprehensive Study on IoT Based Accident Detection Systems for Smart Vehicles," in *IEEE Access*, vol. 8, pp. 122480-122497, 2020, doi: 10.1109/ACCESS.2020.3006887.

[4]. M. Saffarini, R. Saffarini and I. Ishaq, "Smart System to Avoid Car Accidents," *2020 International Conference on Promising Electronic Technologies (ICPET)*, Jerusalem, Palestine, 2020, pp. 22-27, doi: 10.1109/ICPET51420.2020.00013.

[5] W. Rahman, M. R. Ruman, K. Roushan Jahan, M. J. Roni, M. Foyjur Rahman and M. A. Hasnat Shahriar,"Vehicle Speed Control and Accident-Avoidance System Based on Arm M4 Microprocessor," *2020 International Conference on Industry 4.0 Technology (I4Tech)*, Pune, India, 2020, pp. 154-158, doi: 10.1109/I4Tech48345.2020.9102693.

[6]. T. Sangeetha, M. Kumaraguru, S. Akshay, and M. Kanishka, ‘‘Biometric based fingerprint verification system for ATM machines,’’ J. Phys., Conf. Ser., vol. 1916, no. 1, May 2021, Art. no. 012033.

[7]. ] K. Xia, H. Fan, J. Huang, H. Wang, J. Ren, Q. Jian, and D. Wei, ‘‘An intelligent self-service vending system for smart retail,’’ Sensors, vol. 21, no. 10, p. 3560, May 2021.

[8]. C. Xiong, ‘‘Design of intelligent garbage classification bin based on LD3320,’’ in Proc. Int. Conf. Signal Process. Mach. Learn. (CONFSPML), Nov. 2021, pp. 7–10.

[9]. P. Fu, D. Liu, and H. Yang, ‘‘LAS-transformer: An enhanced transformer based on the local attention mechanism for speech recognition,’’ Information, vol. 13, no. 5, p. 250, May 2022.

[10]. J.-W. Hu, B.-Y. Zheng, C. Wang, C.-H. Zhao, X.-L. Hou, Q. Pan, and Z. Xu, ‘‘A survey on multi-sensor fusion based obstacle detection for intelligent ground vehicles in off-road environments,’’ Frontiers Inf. Technol. Electron. Eng., vol. 21, no. 5, pp. 675–692, May 2020.